الجمهورية الجزائرية الديمقراطية الشعبية
REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
وزارة التكوين والتعليم المهنيين
Ministère de la Formation et de l'Enseignement Professionnels

Centre National de l'Enseignement Professionnel à Distance (CNEPD)

المركز الوطني للتعليم المهني عن بعد (م.و.ت.م.ب)

COURS D'ALGORITHME

SÉRIE 03

LES STRUCTURES DE CONTROLE

OBJECTIF PÉDAGOGIQUE : À la fin de cette série, le stagiaire doit être capable de combiner plusieurs structures de contrôle.

PLAN DE LA LEÇON

I- LA RÉPÉTITION :

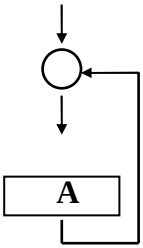
Les ordinateurs (on le sait, sont contrairement aux humains) aptes à répéter de nombreuses fois, sans fatigue apparente, des actions identique ou similaires. Il est donc naturel d'introduire comme premier type de structure de contrôle la répétition sous différentes formes (nous utiliserons à ce sujet les mots ("répétition», «boucle".)

1- Répétition à l'infini :

Un type de répétition qui s'impose immédiatement à l'esprit est la répétition à l'infini, que nous noterons :

Répétition à l'infini
    action (s)

Et qu'illustre l'organigramme



Répéter à l'infini
| A

Exemple :

Considérons, par exemple un ordinateur chargé de surveiller un réseau électrique, et capable d'effectuer l'opération :

«Vérifier la tension du réseau »

La tension doit, en principe, être constamment surveillée : il s'agit donc d'une répétition à l'infini (« à l'infini » signifie en pratique jusqu'à la prochaine panne).

Le programme du calculateur s'écrira :

Répéter à l'infini

Vérifier la tension du réseau ;

Dans la plupart des cas, nous considérons par la suite, les processus entièrement finis, et nous aurons donc que rarement à mentionner la répétition à l'infini. Il convient cependant de ne pas négliger cette structure de contrôle, dont le rôle est capitale dans le domaine des processus temps réels

2- Répétitions contrôlées par des conditions :

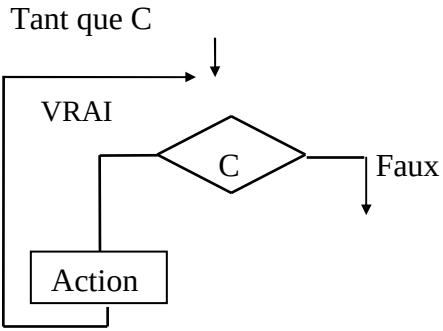
Dans les cas usuels, cependant, on veut répéter une action non pas indéfiniment, mais seulement tant qu'une certaine condition est vérifiée.

On notera une telle opération :

Tant que < condition >
Faire
<action> ;
Fait;

La condition est une expression logique (c'est à dire une expression susceptible de prendre l'une des deux valeurs Vrai ou Faux) dépendant des variables du programme. L'action peut être une suite d'actions.

Il est clair que l'exécution d'une telle boucle n'est susceptible de se terminer que si l'action peut modifier l'un des éléments intervenant dans la condition ou si celle-ci est fausse dés l'origine pour sortir du 'Tant que'.



Faire
Action ;
Fait ;

Exemple :

Calculer la somme des 100 premiers entiers.

Analyse :

Tant que l'entier n'est pas 100 j'additionne
0 + 1 + 2 + 3 +. + 50 + 99

Algorithme somme des 100 premiers entiers :

Début

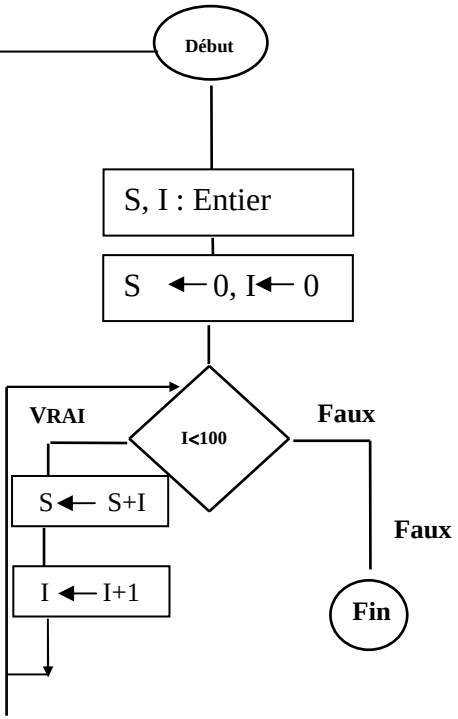
S, I: Entier;
S: = 0; I: = 0;

Tant que I < 100

Faire
S = S + I ;
I = I + 1 ;

Fait:

Fin.



Commentaires sur l'algorithme :

- Je donne un nom significatif à mon algorithme
- Je déclare les variables entières S et I :
- J'affecte la valeur zéro aux variables S (somme) et I (entier)

Et je place ma structure de contrôle tant que I < 100 Faire ici je dresse un

Tableau pour effectuer la trace de cette structure.

S	I	Tant que I < 100	actions
0	0	0 < 100 vrai	S = S + I = 0 + 0 = 0 I = I + 1 = 0 + 1 = 1
0	1	1 < 100 vrai	S = S + I = 0 + 1 = 1 I = I + 1 = 1 +1 = 2
1	2	2 < 100 vrai	S = S+ I = 1 + 2 = 3 I = I + 1 = 2 + 1 = 3
.	.	.	.
.	98	98 < 100 vrai	S = S + I = I = I + 1 = 98 + 1 = 99
.	99	99 < 100 vrai	S = S + 1 I = I + 1 = 99 + 1 = 100
.	100	100 < 100 faux	Arrêt de la boucle

Une autre forme de répétition voisine de la précédente est la boucle :

Répéter :

| Action ;

Jusqu'à condition ;

Faire

| Action ;

Jusqu'à condition ;

Qui effectue action et la répéter jusqu'à ce que condition soit vraie :

Exemple précèdent

Algorithme somme des 100 premiers entiers

Début

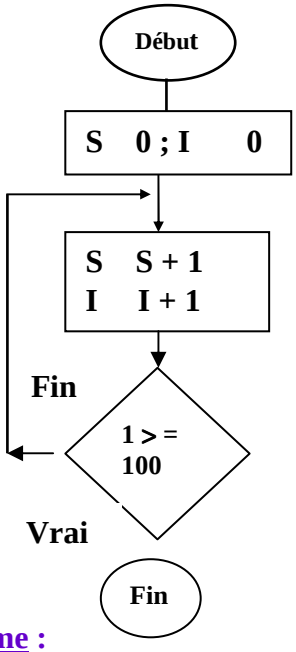
S, I: Entier;
S: = 0; I: = 0;

Répéter

S ← S + I ;
I ← I + 1 ;

Jusqu'à I > = 100

Fin.



Commentaires sur l'algorithme :

Nous pouvons dérouler la trace de cet algorithme de la même façon que le précèdent, par conséquent nous allons arriver au même résultat, sauf, qu'ici, à la différence du cas précèdent, action est toujours exécutée au moins une fois, quelque que soit la valeur initiale de condition (I >=100)
En résumé on peut dire que :

Répéter
A;
Jusqu'à C

est équivalent à:

A;
Tant que C
Faire
A ;
Fait

II- LE CHOIX:

À partir d'un problème posé, comme la résolution de l'équation ax + b = 0, a et b réel, b # 0. Nous allons définir la structure du choix. Nous pouvons analyser l'équation de la façon suivante :

Si a = 0, alors l'équation n'a pas de racine sinon la racine est la valeur de x = - b/a.

L'exécution des instructions par la machine va dépendre des valeurs de a et b fournies par l'utilisateur .Il faut donc prévoir dans l'algorithme tous les cas possibles.

Ainsi, on peut dire que l'algorithme va apparaître ici comme une manière de prévoir toutes les suites d'instructions susceptibles d'être exécutées par la machine.

Pour représenter notre algorithme, nous utilisons un outil qui permet de comparer 2 objets de même nature pour cela, nous allons introduire une instruction conditionnelle.

1- L'alternative :

La forme la plus simple d'une situation de choix est l'alternative, où deux décisions sont possibles et où l'arbitrage dépend d'une condition vraie ou fausse

Nous noterons l'alternative de la façon suivante :

Si <condition >

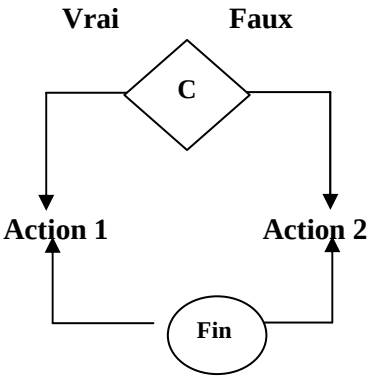
Alors

<action 1 >

Sinon

<action 2> ;

FSI ;



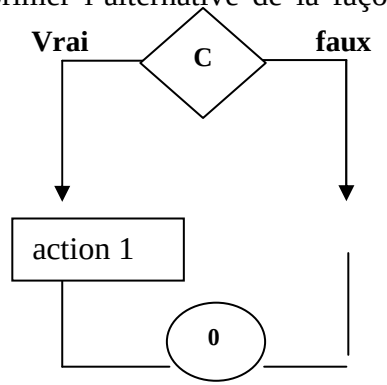
Cette action s'exécute de la manière suivante :

- Si la condition définie dans <Condition> est vérifiée, il faut exécuter l'action (ou le groupe d'actions) définie dans <action 1> puis les actions qui suivent le FSI.
- Si cette condition n'est pas vérifiée, il faut exécuter l'action (ou le groupe d'actions) définie dans < action 2> puis les actions qui suivent le FSI.

Dans certains cas, on peut exprimer l'alternative de la façon suivante :

Si <condition>
alors
action 1 ;

FSI :



Dans ce cas, l'action à effectuer si la " condition " est fausse (c'est à dire l'action qui suit le sinon) est une action nulle (ne rien faire), on convient d'omettre le sinon et ce qui le suit.

Remarque : La dernière action vient juste avant le sinon ne doit pas se terminer par un point virgule ' ;'

Exemple 1 :

```
SI  WILAYA = 16
    Alors
        Ecrire (ALGER)
    Sinon
        Ecrire (' WILAYA autre qu'Alger')

FSI;
```

Exemple 2 :

```
SI   DELTA < 0
|    Alors
|        Ecrire ('Pas de solution') ;
FSI ;
```

Exemple 3:

```
SI    A<B
|     Alors
|           MIN : = A
|     Sinon
|           MIN : = B
FSI;
Ecrire (MIN);
```

Exemple 4:

Résolution d'une équation du premier degré algorithme :
équation ax + b = 0

```
Début
|    a,b,x : réel ;
|    Lire (a, b);
|    Si  a = 0
|    |    Alors
|    |         Écrire (" l'équation n'a pas de racine ");
|    |    Sinon
|    |         x = - b/a
|    |         Écrire (" la racine x = ") ;
|    |         Écrire (x) ;
     FSI;
Fin.
```

2- Le choix multiple :

Souvent, on veut choisir entre non plus deux, mais plusieurs possibilités, on écrira alors:

Décider entre

< condition $_1$ > :< action $_1$ > ;
< condition $_2$ > : <action $_2$ > ;

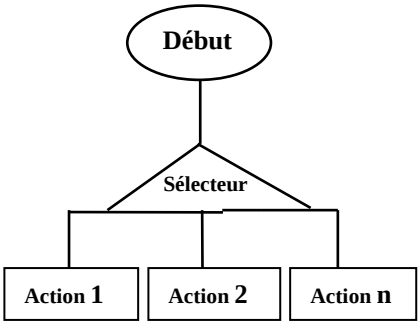
- - - - - - - - - -
- - - - - - - - - -- - - -

<condition $_n$ > : <action $_n$ > ;

Autrement

< Action $_0$ > ;

FIN



Cet ordre nous mènera à l'exécution d'une action $_i$ lorsque la condition $_i$ soit vraie ; ou à défaut, si aucune des conditions $_i$ $_{(i = 1, 2, . . ., n)}$ n'est vraie, on exécute action $_0$.

L'algorithme précédent est équivalent a l'algorithme suivant :

Si <condition $_1$>
alors <action $_1$>
Sinon
Si <condition $_2$>
alors <action $_2$>
Sinon
Si - - - - - - - - - - - - -
Sinon
Si <condition $_n$ >
alors <action $_n$ >
Sinon <action $_0$>

FSI

APPLICATION :

Algorithme déterminer les nombres correspondant.

Début

N : Entier ;
Lire (N); {un nombre entrer entre 1 et 12}

Décider entre

N = 1 : écrire (« c'est le mois de Janvier ») ;
N = 2 : écrire (« c'est le mois de Février ») ;
N = 3 : écrire (« c'est le mois de mars «) ;
N = 4 : écrire (« c'est le mois d'Avril ») ;
" " "
" " "
" " "
N = 11 : écrire (« c'est le mois de Novembre ») ;
N = 12 : écrire (« c'est le mois de Décembre ») ;

Autrement

Ecrire (vous avez fait une erreur);

Fin.

III- IMBRICATIONS DE STRUCTURES ALTERNATIVES ET RÉPÉTITIVES :

Dans ce cas, aucune action nouvelle n'est introduite, les algorithmes qui y sont décrits utilisent une combinaison de structures alternatives et de structures répétitives.

Ainsi on peut avoir :

```
Tant que  1ère condition
        Faire
            Si    2ème condition
                  alors
                        action 1;
                  Sinon
                        action2 ;
            Fsi
        Fait
Ou
Si   1ère condition
    Alors
        Tant que   2ème condition
                   Faire
                        action 1 ;
                   Fait
    Sinon
        action  2 ;
Fsi ;
```

Exemple :

Apres avoir corrigé les copies d'un examen, un professeur veut déterminer le nombre de notes supérieures à 10 et le nombre de notes supérieures à 18.

Le nombre total de notes est connu.

Analyse : Nombre : nombre de notes

Note : note élève

SUP 10 : nombre de notes > 10

SUP 18 : nombre de notes > 18

Tant qu'il y a des notes à lire, je teste si la note est supérieure à 10 alors je comptabilise les notes supérieur à la moyenne dans SUP_10 et Si elle est supérieure à 18, dans ce cas je comptabilise les notes supérieure à 18 dans SUP_18.

Algorithme : corrigé d'examen.

```
Début
    Nombre, SUP_10, SUP_18 : entier ;
    Note : réel ;
    lire (Nombre) ;
    SUP_10 : =  0 ; SUP_18 : =  0 ;
    Tant que  nombre  > 0
            Faire
        lire (note) ;
            Si  note > 10
                Alors
                    SUP_10 : =  SUP_10 + 1 ;
                    Si  note > 18
                        Alors
                            SUP_18 : =  SUP_18 +1 ;
                    Fsi ;
            Fsi;
                Fait
Écrire (« le nombre de notes supérieures  à 10 est ", SUP_10) ;
Écrire (" le nombre de notes supérieures à 18 est ", SUP _18) ;
Fin.
```

IV-L'ENCHAINEMENT :

L'enchaînement est une structure de contrôle tellement naturelle que l'on a parfois tendance à oublier son caractère fondamental.

L'idée de l'enchaînement est que si les ordres Action 1 et Action 2 spécifient des actions réalisables la notation action $_1$; action $_2$. En indique une autre, obtenue en faisant suivre l'exécution de Action 1 puis celle de Action 2.

Action 1 ; action 2 ; équivalent à : Action 1 ;
Action 2 ;

La notation utilise le point-virgule (;) pour représenter l'enchaînement. Elle permet d'exprimer l'enchaînement de plusieurs Actions.

Action 1 ;
Action 2 ;
Action 3 ;
"
"
Action n-1 ;
Action n ;

L'intérêt principal de cette notion d'enchaînement est que l'instruction composée résultante joue le même rôle qu'une instruction simple.

Nous écrirons donc par exemple :

Tant que condition
Faire
| Action 1 ;
| Action 2 ;
| Action 3 ;
Fait

Où le trait vertical est la pour rappeler cette homogénéité du bloc Action 1 ; Action 2 ; Action 3.

À titre d'exemple des applications de l'enchaînement. Reprenons notre exemple de la somme des 100 premiers nombres entiers

Algorithme somme des 100 premiers nombres entiers

Début

S, I: entier
S: = 0 ; I: = 0;
Tant que I < 100

Faire
S: = S + I
I: = I + 1
Fait ;

Fin.

Vous voyez qu'ici, le bloc d'instructions est constitué de deux instructions
(S := S + I ; I := I + 1), ce bloc doit être répéter tant que I < 100.

RÉSUMÉ :

Dans cette leçon, nous avons vu les structures de contrôle en algorithmique tel que la répétition qui s'exprime par la boucle Tant que <Condition> Faire <Actions> Fait et la boucle Répéter <Actions> Jusqu'à <Condition>, le choix qui s'exprime par Si <Condition> Alors <Action1> Sinon <Action2> Fsi et enfin l'enchaînement d'actions qui se traduit par une suite d'actions séparées par des points virgules.

Nous avons vu aussi qu'une combinaison de plusieurs structures de contrôle est possible.

EXERCICES D'APPLICATION :

EXERCICE N° 01 :

Écrire un algorithme qui permet d'imprimer les carrés des entiers impairs de 1 à 100 et leurs sommes.

EXERCICE N° 02 :

Ecrire un algorithme qui permet de calculer la valeur de factorielle N.

EXERCICE N° 03 :

Calculer le nième terme de la suite de FIBONACCI définie par :

$U_1 = 1, U_2 = 2$ et $U_n = U_{n-1} + U_{n-2}$ (n > = 3)

EXERCICE N° 04:

Mer AKLI a acheté une certaine quantité d'un produit dont le prix unitaire est de 50DA.

Si le prix à payer est inférieur à 200 DA, on doit ajouter 25 DA de frais de transport.

Editer le montant de la facture à adresser à Mer AKLI.

EXERCICE N° 05:

On donne deux nombre entiers X et Y, éditer la plus grande valeur en établissant un algorithme appelé GRAVAL.

EXERCICE N° 06:

Apres avoir corrigé les copies d'un examen, le professeur veut déterminer le nombre de notes supérieures à 8 et le nombre de notes supérieures à 15.

Le nombre total des notes doit être connu (c'est a dire lecture du nombre)

CORRECTION DES EXERCICES :

EXERCICE N° 01 :

a) ALGO carré1;

Début

I, S: entier

S: = 0

Pour I:= 1 à 99 Par de 2

Faire

écrire (I*I);

S: = S + (I*I);

Fait;

Ecrire (‘ La somme des carrés = ‘, S);

Fin.

b) ALGO carré 2;

Début

I, S: entier

S: = 0, I: =1

Tant que I <= 99

Faire

Écrire (I*I);

S: = S + (I*I);

Fait;

Ecrire (‘ La somme des carrés = ‘, S);

Fin.

EXERCICE N° 02 :

```
ALGO FACT ;
   N, F, I ; entier ;
  Début
    lire (N);
    F:=1;
    SI  N>0
       Alors
            Pour  I:=1 à N
                 Faire
                      F:=F*I;
                 Fait;
      Écrire ('Factorielle =', F);
    Fsi
  Fin.
```

Vous voyez qu'ici on a utilisé une structure imbriquée où la structure conditionnelle (Si/Fsi) contient la structure de contrôle (pour). On pouvait remplacer la boucle (pour) par la boucle (tant que), dont voici la partie de son algorithme : Si N>0

```
Alors
        I:=1
        Tant que  I <= N
                Faire
                    F: =F*I;
                    I: =I+1;
                Fait;
        Écrire ('Factorielle =', F);
      Fsi;
```

EXERCICE N° 03:

ALGO FIBIONACCI;

I, N,U,U_1,U_2 : entier ;

Début
lire (N);
Si N >= 3
Alors
U_1 := 1 ;
U_2 := 2 ;
Pour I := 3 à N
Faire
U_1 := U_1+ U_2
U_1 := U_2 ;
U_2 := U
Fait ;
Écrire (‘Le nième terme de la suite =‘, U) ;
Sinon
Écrire (‘Vous faite erreur, n est à 3’) ;
Fsi ;
Fin.

L’application est simple, il fallait lire n>3 sinon l’algorithme vous fait afficher ERREUR. Dans le cas affirmatif (jusqu'à n>=3) ,il y a une boucle qui permet de varier de 3 à N entrée au clavier , dans le corps de cette boucle ou calculé le prochain terme U = U_1+ U_2, puis on affecte U_2 dans U_1 et V dans U et répété le traitement jusqu'à I soit égal à N.

EXERCICE N° 04 :

ALGO AKLI ;

Début

Q, PBRUT, PNET : réel ;
Lire (‘ Donner la quantité =‘ Q) ;
PBRUT := Q*50 ;
Si PBRUT > = 200
Alors PNET := PBRUT
Sinon PNET := PBRUT+25

Fsi;
Écrire (‘Le montant = ‘, PNET);
Fin.

La quantité de produit acheté par Mer AKLI est représentée par une valeur (un nombre réel) disponible sur le clavier (lire(Q)) pour la représenter en mémoire centrale, on utilise une variable réelle Q. On utilise également des variables réelles PBRUT (valeur intermédiaire) et PNET (montant de la facture) qui sera édité.

EXERCICE N° 05 :

ALGO SUPER ;

Début

X, Y, PG : entier ;
lire (‘ Donner la valeur de X’, X);
lire (‘ Donner la valeur de Y’, Y);
Si X >Y
Alors PG := X
Écrire (‘la valeur X = ‘ X,’est supérieure à Y’, Y)

Sinon
Si X<Y
Alors

Écrire (‘la valeur Y = ‘, Y, c’est supérieure à X ‘, X)

Sinon

Écrire (‘les valeurs X et Y sont égales)

Fsi;

Fsi;

Fin.

EXERCICE N° 06:

ALGO NOTES;

Début

SUPER8, SUPER 15 : entier;

Nombre : réel ;

Écrire (‘Indiquer le nombre de notes à traiter’);

lire (nombre);

SUPER8 :=0; SUPER15 := 0;

Tant que Nombre > 0

Faire

Écrire (‘Donner une note’);

Si Note > 8

Alors

SUPER8 := SUPER8 +1

Si Note > 15

Alors

SUPER15 := SUPER15 +1

Fsi;

Fsi;

Fait;

Fin.